Weekly Report(July.8.2019-July.14.2019)

Shanghai Jiao Tong University
1747836307@qq.com
Shaopu Rui

Abstract

This week I learned to use tensorbardX and continued to train AlexNet.

1 Lessions from last week's questions

- First, training data is more important than a network. We should always rebuild the network structure instead of resizing the input data.
- Then I'd like to show my flowchart here. I am sorry to forget it last week.

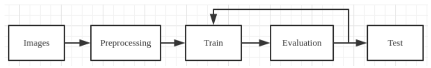


Figure 1: System Flow Chart

2 TensorBoardX

With my buddy's suggestion, I use TensorBoardX to record my training result this time.

Store the result like this:

```
from tensorboardX import SummaryWriter
writer = SummaryWriter(comment="train")
writer.add_scalar('train', loss.item(), t)
```

And run it to show them by this:

```
tensorboard --logdir runs/
```

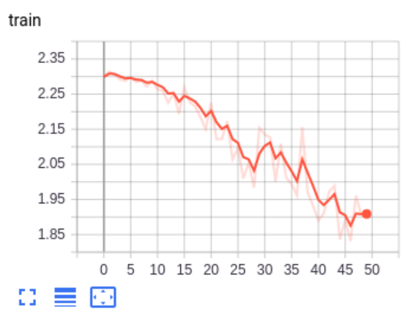
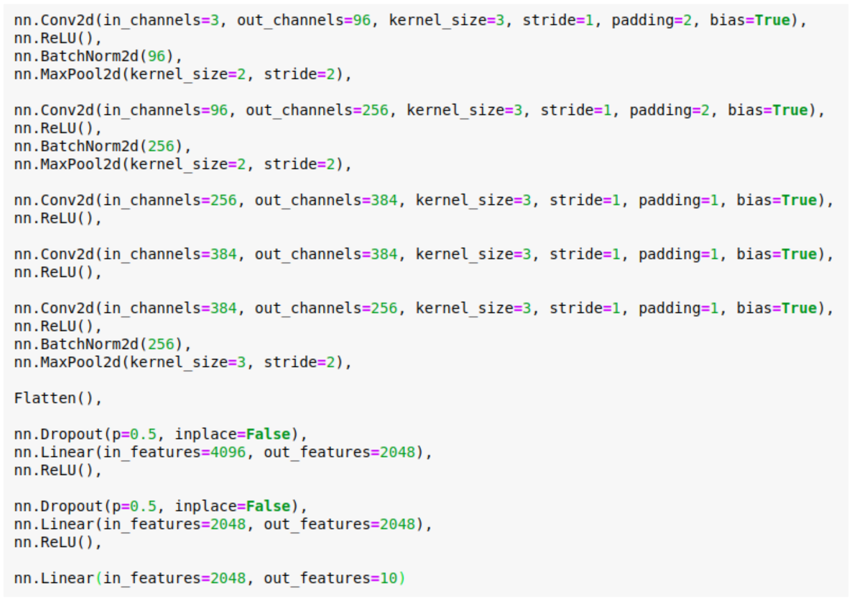


Figure 2: First loss line

3 AlexNet

I rebuilt the structure of AlexNet. It didn't take me too much time so I spend more time to train and try to get some expected result.

Figure 2 shows the new network. Combined to the origin one, I just changed some parameters, for example like kernel size, to adapt to input dataset and add Batch Normalization to avoid saturation.



```
nn.Conv2d(in_channels=3, out_channels=96, kernel_size=3, stride=1, padding=2, bias=True),
nn.ReLU(),
nn.BatchNorm2d(96),
nn.MaxPool2d(kernel_size=2, stride=2),

nn.Conv2d(in_channels=96, out_channels=256, kernel_size=3, stride=1, padding=2, bias=True),
nn.ReLU(),
nn.BatchNorm2d(256),
nn.MaxPool2d(kernel_size=2, stride=2),

nn.Conv2d(in_channels=256, out_channels=384, kernel_size=3, stride=1, padding=1, bias=True),
nn.ReLU(),

nn.Conv2d(in_channels=384, out_channels=384, kernel_size=3, stride=1, padding=1, bias=True),
nn.ReLU(),

nn.Conv2d(in_channels=384, out_channels=256, kernel_size=3, stride=1, padding=1, bias=True),
nn.ReLU(),
nn.BatchNorm2d(256),
nn.MaxPool2d(kernel_size=3, stride=2),

Flatten(),

nn.Dropout(p=0.5, inplace=False),
nn.Linear(in_features=4096, out_features=2048),
nn.ReLU(),

nn.Dropout(p=0.5, inplace=False),
nn.Linear(in_features=2048, out_features=2048),
nn.ReLU(),

nn.Linear(in_features=2048, out_features=10)
```

Figure 3: AlexNet Structure

The visualization of the training process is in Figure 3. Definitely tensorboardX makes it more evident to know how the training works.

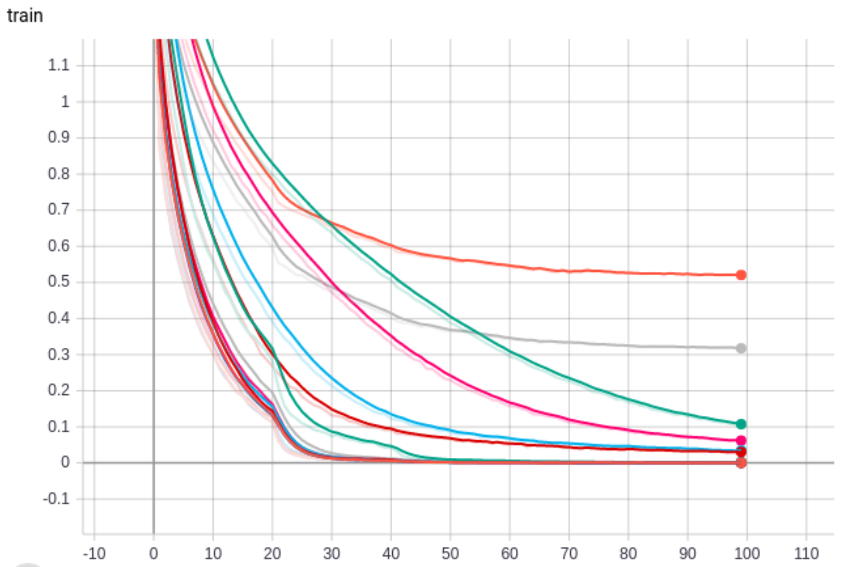


Figure 4: AlexNet Structure

I trained different models with different hyper parameters and different optimizers. The real output is too much, I will just show some simple data here.

- The hyper parameters and their result of different models are showed below. Since it's the beginning, I set the learning rate start from 0.01 and multiple 1/3 for every 20 epoches.

Table 1: learning_rate=0.01, and multiples 1/3 for every 20 epoches

Batch Size	**Momentum**	**Accuracy**
64	0.9	87.59%
64	0.95	88.08%
128	0.9	88.09%
128	0.95	87.56%

- I wonder whether learning rate is the critical factor making accuracy so high. So I keep it the same. The result is in table2. It seems that I am wrong.

Table 2: learning_rate=0.01, momentum=0.95

Batch Size	**Accuracy**
64	87.44%
128	87.61%
256	87.27%

- I still tried another optimizer–Adadelta. Since Adadelta is said to train the network faster and more precisely. The Hype parameters and results are in table3. It seems not so good. In my opinion, learning rate may be a important factor. I will show it in next report.

Table 3: learning_rate=0.01, optimizer=Adadelta

Batch Size	**Accuracy**
128	74.66%
256	72.63%

4 Plan

- Just as I said in the last report, I will start to use pre-trained VGG to build model and fine tune the model with cifar10. Time is limited, I will do as more as possible.